I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N°1878

CACHING AND SCHEDULING MECHANISMS FOR H.264 VIDEO FLOW DELIVERY OVER DISCONTINUOUS COVERAGE WIRELESS NETWORKS

AZZA JEDIDI , FRÉDÉRIC WEIS

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Caching and scheduling mechanisms for H.264 video flow delivery over discontinuous coverage wireless networks

Azza Jedidi [*] , Frédéric Weis [**]

Systèmes communicants
Projet ACES

Publication interne n1878 — March 2008 — 15 pages

Abstract: The past few years have witnessed the deployment of a wide variety of multimedia applications over wireless networks. As a consequence, mobile users are demanding fast and efficient connectivity, especially concerning start-up delays and flow quality. Wireless networks aim at providing an everywhere coverage without spending prohibitive deployment costs. Thus, the coverage of each access point is extended with detriment to the mean throughput of the radio cell. The coverage is continuous, but radio conditions met by users are not homogeneous. Actually, data rate is varying according to user mobility. In the context of streaming applications, intermittent high rate availability may lead to service disruption, especially when user's number increases. Thus, it is essential to design a network architecture able to efficiently deliver video flows to mobile users. In our works, a new equipment, called **network cache**, is introduced. The latter performs efficient flow caching and scheduling, in order to guarantee service continuity. The simulation shows that the proposed architecture gives satisfying service start-up delays and improves system scalability.

Key-words: Mobility, logical discontinuous coverage wireless network, caching, scheduling, scalability, start-up delay, resource management.

(Résumé : tsvp)

* This paper presents results from a collaboration between INRIA ACES research team and the MAG project from Alcatel-Lucent Bell Labs

[*] azza.jedidi@irisa.fr
[**] frederic.weis@irisa.fr

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Mécanismes de gestion de caches et de distribution de flux H.264 pour des réseaux à couverture discontinue

Résumé : Les réseaux sans fils les plus récents proposent des services et applications multimédias de plus en plus évoluées. Face à une telle offre, les utilisateurs mobiles deviennent de plus en plus exigeants, en particulier en termes de qualité des flux reçus et de rapidité du démarrage du service.
Les réseaux sans fils visent à offrir une couverture omniprésente et ce à des coûts raisonnables. L'idée est d'étendre la couverture de chaque point d'accès, au détriment du débit moyen de la cellule. La couverture radio est continue, néanmoins les conditions radio ne sont pas homogènes pour tous les utilisateurs. Le débit de données dépend en effet de la mobilité des utilisateurs.
Dans le cadre particulier des applications de *streaming*, une baisse ponctuelle de débit peut amener à une interruption du service, en particulier lorsque la densité d'utilisateurs augmente. Il s'avère donc important de concevoir une architecture réseau capable de distribuer efficacement les flux vers les utilisateurs mobiles. Dans nos travaux, on définit un nouvel équipement réseau, appelé cache réseau (*network cache*). Il permet de stocker ,de conserver et d'ordonnancer efficacement les flux afin d'assurer la continuité du service.
Les simulations montrent que l'architecture ainsi obtenue apporte des résultats satisfaisants en termes de latence de démarrage du service et de passage à l'échelle du système.

Mots clés : Mobilité, réseau à couverture logiquement discontinue, mise en cache, ordonnancement, mise à l'échelle, latence de démarrage de service, gestion de ressource.

1 Introduction

Wireless data transfers play an important role in nowadays multimedia communications, especially thanks to the increase of available bandwidth and the growing needs for mobility. As a consequence, users are demanding fast and efficient ubiquitous connectivity. They require «any-time any-where» services, without delays or disruptions. Therefore, new network infrastructures had to be designed in order to guarantee ubiquitous coverage and to face wireless network constraints, in terms of bandwidth, error rates and other perturbations linked to environment heterogeneity.
In this context, the logical discontinuous coverage wireless network model has been proposed. This new model consists of a set of access points, *i.e.* antennas around of which are defined the radio cells. Those antennas are discontinuously spread on the network area, thus providing a «many-time many-where» service. Actually, the idea of coverage discontinuity brings two major advantages. First, as it implies the use of a fewer number of access points, the architecture deployment will be cheaper. Second, radio cells disjunction hypothesis simplifies the radio frequency band management and avoids interference problems. The mechanisms defined in the framework of discontinuous radio coverage network model can be easily applied to the context of continuous radio coverage networks, where bandwidth is highly varying. High bandwidth areas are similar to coverage areas, and low bandwidth areas are similar to out of coverage areas. The main idea is to discriminate terminals by their radio conditions in order to select the data to be sent. In fact, terminals are supplied with data when they cross high bandwidth areas (*transfer areas*). And, low bandwidth areas are dedicated to detecting terminals presence (*presence areas*). Our research team has proved through simulations that avoiding data transfer in low bandwidth areas enhances the throughput use over the network and improves system scalability [1].
This new logically discontinuous vision of the network makes this model fit to 3rd generation networks, where coverage is continuous but with important bandwidth fluctuations, as well as 4th generation networks, where radio coverage could be physically discontinuous. Even if this model simplifies network deployment, the connectivity intermittent induces important challenges in order to avoid service disruption. Thus, terminals have to take advantage of the high bandwidth available when crossing a transfer area in order to store some part of the multimedia flow on their own caches, and to consume it when crossing presence areas.
Our works focus on logical discontinuous coverage wireless networks, and aim at providing a multitude of services. In this paper, our main goal consists in designing a scalable unicast streaming service, while reducing the start-up delays.
On the one hand, streaming servers are usually located in «classical» IP networks. Such networks are submitted to many constraints, like bandwidth variations. On the other hand, mobile terminals evolve in wireless logical discontinuous coverage areas, constituted of several access points. Each access point defines a high bandwidth area in each radio cell. The main idea we propose in the context of such networks, is to design an intermediate equipment that «catches» the video flows sent by the content server, and then efficiently distributes them to mobile terminals in the network. This equipment is called the **network cache** (see figure 1).
In a previous version of our works, the network cache acted as a temporary storage memory. Actually, it temporarily stored the part of the flow to be delivered to the mobile terminal, while the latter

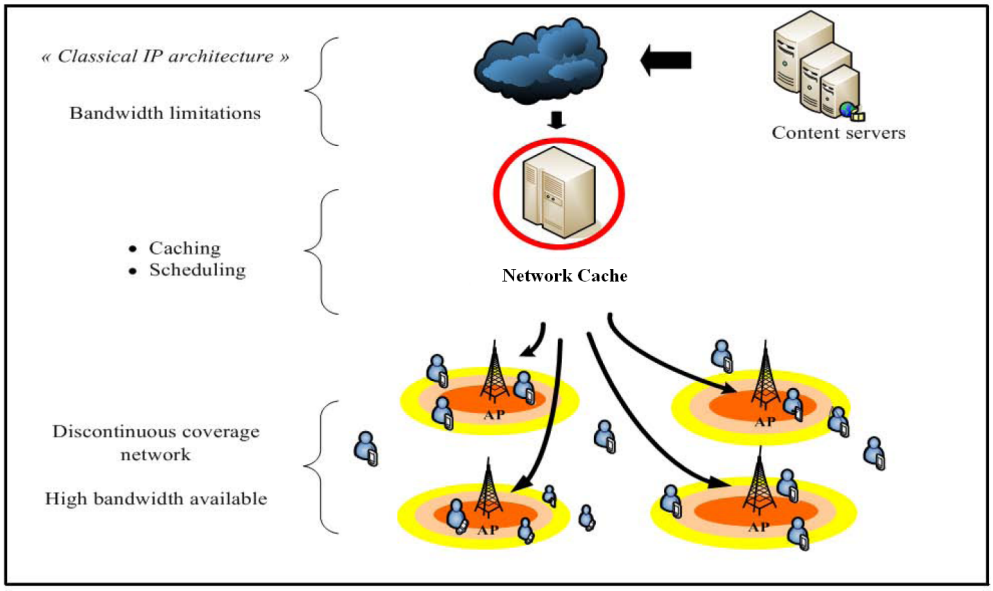

Figure 1: Logical discontinuous coverage wireless network

crossed a presence area. Then, when the terminal entered a transfer area, the network cache rapidly delivered the stored flow to him, taking benefit of the high bandwidth available. More precisely, the network cache was provided with a set of buffers. Each buffer was related to a single streaming request and contained the part of the flow to be sent to the corresponding mobile terminal [1]. The use of buffers considerably enhanced the network scalability. In fact, it allowed users, crossing transfer areas, to take advantage of the high bandwidth available, in order to rapidly dispose of the requested flow. Flow delivery was no more subject to the bandwidth limitations at the content server level, as the flows were already stored in the network cache.

Terminals also were provided with buffers to store some parts of the flow in advance (when the the terminal crosses transfer area), in order to consume it while crossing out of coverage areas and, thus, avoid service disruption.

Moreover, the network cache played a scheduling role. It classified flows in priority queues, depending (1) on terminal positions in the network and (2) on terminal buffers content. The scheduling policy favored terminals which were in transfer areas, as they will rapidly receive their flows thanks to the high bandwidth available. It also favored terminals whose buffer level was under a certain threshold, in order to avoid service disruption.

We proved through simulations that, thanks to the network cache buffers and the scheduling policy, the scalability of the network was considerably enhanced as the service disruption was avoided for a larger terminal density [1].

Nevertheless, buffers are temporary storage memories. So, the flow stored in a buffer is used by only

one user, and could not benefit to users who ask for the same file later. In this paper, we propose to replace the buffers in the network cache with caching mechanisms. Actually, a copy of any requested video flow will be kept in the network cache, after the end of the streaming request. Later requests to the same video flow will be accessed faster as the flow is already stored in the cache network, nearer to the terminal than if it was in the content server. The delivery is no more subject to «classical» IP network bandwidth constraints (*i.e.* bandwidth constraints in the path between the content server and the network cache). Buffers, in the terminal, will also be replaced by a cache structure in order to store sufficient parts of the flow and to avoid service disruption.
Moreover, we propose to bring a more efficient scheduling mechanism in the network cache, taking into account some properties of the video flow in addition to terminal position and cache level. As H.264 flows are more and more used for nowadays mobile video applications, we chose to consider H.264 video flows for our streaming service.
The remainder of this paper is organized as follows: first we make a brief study of the main video flow caching and scheduling mechanisms. Then, we present the mechanisms we choose in the context of our network. After that, the targeted architecture is detailed. Finally, the performances of our solution are evaluated.

2 Related works

2.1 Video flow Caching mechanisms

Video flow caching algorithms depend on different criteria linked to the flow itself (like the flow size or encoding) or related to the network (like the available bandwidth) or even to the terminals. In fact, the choice of a caching mechanism is tightly coupled with the homogeneity or heterogeneity of terminals.

2.1.1 Video flow Caching mechanisms for homogeneous terminals

Several existing caching algorithms are based on the hypothesis that terminals are homogeneous. It means that they have identical, or at least similar, performances and configurations. Thus, any video object is supposed to meet the capabilities of all terminals, in terms of bandwidth and format. Those algorithms focus on how to efficiently choose the part of the flow that should be cached.

Sliding-interval caching This method is based on caching a sliding interval of a video flow to exploit prospective sequential access to that flow. In fact, the first request recuperates the flow from the server and incrementally stores it into a proxy cache. The second request, if it occurs after a short delay, may then access the cached flow in the proxy, and flushes it after the access [2]. This approach may be applied for more than two requests, arriving in a short time. In that case, a set of adjacent intervals can be grouped to form a «run». The cached flow is flushed when the last request is satisfied.

Prefix Caching This approach consists of caching the prefix of a video flow, *i.e.* its initial portion in a proxy. Since a client requests a video flow the proxy immediately delivers to him this flow prefix. Simultaneously the proxy fetches the suffix (*i.e.* the remaining part of the flow) from the server and transmits it to the client [3]. This approach reduces significantly the start-up delay as the proxies are generally closer to the clients than the media server .

Segment-based distance-sensitive caching This method proposes to group the frames of a media object into variable-sized segments. The segment length increases exponentially. Thus, the size of segment i is 2^{i-1}, which consists of frames [2^{i-1}, 2^{i-1}+1 ... , 2^{i}-1]. As a consequence, the last segment length is half the cached portion length [4]. Such segmentation is especially interesting regarding the cache replacement policy. Indeed the latter will delete segments starting from the end of the cached flow. So, it will rapidly discard as big chunks as needed (see figure 2).

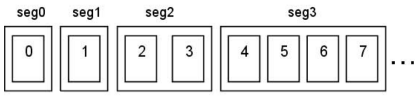

Figure 2: Segment-based distance-sensitive caching

Rate-Split Caching This approach consists of partitioning the video flow along the rate axis: the upper part is cached in the proxy, while the lower part remains stored at the video server (see figure 3) . It is especially interesting in the context of VBR streaming, since only the lower part, which is of a nearly constant rate, is delivered through the backbone network [5].

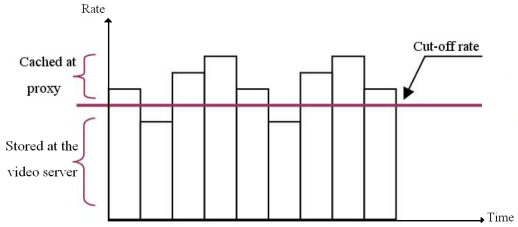

Figure 3: Rate-Split Caching

2.1.2 Video flow Caching mechanisms for heterogeneous terminals

In some cases terminals asking for the same video flow may have quite different requirements. They may need different streaming rates or encoding formats. A solution widely adopted in commercial streaming systems is to produce replicated streams of different rates and formats. Nevertheless, this approach is very demanding in terms of bandwidth and storage. An other possibility is to transcode

flows from one form to another of a lower rate or a different encoding format in an on demand way [6]. The transcoding induces intensive computation overhead, which makes this approach not scalable. A more efficient approach is the layered encoding and transmission method. Layer coders compress the flow in several layers. The base layer contains the data representing the most important elements of the flow. It represents the video flow in its minimal quality. Thus, it is the most significant layer. The other layers are hierarchically added to refine progressively the flow quality [7]. Depending on its capabilities, the terminal will subscribe to a set of cumulative layers to reconstruct the stream.

2.2 Video flow scheduling mechanisms

In the context of logical discontinuous coverage wireless networks, video flow scheduling may become a very hard task as it is subject to a multitude of constraints, such as bandwidth limitation and asymmetry, important error rates and the high probability of service disruption due to client mobility.

Multirate wireless fair queuing (MR-FQ) algorithm consists in transmitting the video flow at different rates depending on the channel conditions [8]. This algorithm takes into consideration both time and service fairness. This algorithm increases the overall system throughput and guarantees fairness and bounded delays.

Layered Quality Adaptation algorithm consists in adding and dropping layers of a layer encoded video stream to perform smooth flow adaptation [9]. The main goal of the algorithm is to fit available bandwidth changes, without rapid and disturbing variations in quality. Actually, congestion control changes the transmission rate rapidly, over several round-trip time, whereas, hierarchical encoding permits smooth video quality adjustment, over long periods of time. The difference between both timescales is obtained thanks to data buffering at the receiver, which smooths the rapid variations in available bandwidth and allows an almost constant number of layers to be played.

Buffer Sensitive Bandwidth Allocation algorithm recommends that a client maintains sufficient video frames at the buffer in order to improve the system adaptability and minimize the impact of overloading [10]. The video playback starts only after reaching a determined buffer level. Based on the play rate of a video, the system calculates the buffer playback duration. The main idea of the method is to allocate the available bandwidth at a base station, which will serve the concurrent requests in the cell, based on their playback buffer durations.

2.3 Conclusions

Many caching algorithms could be adopted in order to design the network cache. In our works, the considered terminals are supposed to be identical. However, the constraints brought by logical coverage discontinuity can be assimilated to those linked to terminals heterogeneity. In fact, homogeneous terminals will need to receive the same file in different sizes, because they will be subject to different constraints, in terms of terminal cache level and available bandwidth. Moreover the

targeted video flows are H.264 SVC (scalable video coding) ones. Those flows are layer encoded, which means that they are constituted of a base layer that represents the flow in a minimal quality, and some enhancement layers that could be hierarchically added to the base layer in order to enhance its quality. In our study, we use a model for the H.264 SVC video flows with a base layer and two enhancement layers, as shown in figure 4. The layered caching approach seems to be the most appropriate caching strategy in the context of such flow model. The layer encoded flow model that we

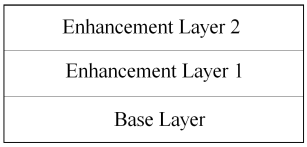

Figure 4: Layer encoded flow model

adopted suggests the quality adaptation as the most appropriate scheduling approach. Moreover, this approach favors system scalability. Our scheduling algorithm, presented in the next section, is also inspired by the Buffer Sensitive Bandwidth Allocation strategy, as we took into account the terminal cache level, when the user enters in the radio cell.

3 H.264 SVC flow delivery over a logical discontinuous coverage wireless network

The key element for video flows delivery in the designed network is the network cache. This equipment is mainly made of two modules: the cache and the scheduler.
The cache is divided into three parts, each of them corresponds to the memory storage of one layer. Thus, for each flow, each layer is stored in the corresponding part of the cache. And, these parts are provided with a mapping table in order to match the memory cells with the corresponding flow identifiers. A similar cache structure is adopted for terminal caches.
The goals of the scheduler are (1) to guarantee the continuity of the streaming service and (2) to shorten the start-up delay. To aim that target, it has to efficiently distribute the video flows to the terminals.
In the previous version, the scheduler dynamically classified the video flows in priority queues, taking into account the terminal position (*i.e.* transfer area / presence area) and its buffer filling level. The streaming service was started when the buffer reached a «start-up buffer level» that corresponds to a playback duration of 30 seconds.
Now, the scheduler classifies the flow layers separately. In fact, it considers the hierarchical order of layers, in addition to terminal position and cache level. As a consequence, the streaming service starting is only linked to base layer caching. Since the equivalent of 30 seconds of playback duration is cached in the terminal, the video playback starts. Moreover, as the scheduling privileges the base

layer transmission, enhancement layers could be delayed or even omitted in order to guarantee service continuity. In fact, service continuity in such a flow model is equivalent to base layer streaming continuity. Actually, the level of terminal cache filling-up necessary to streaming start-up is defined by making a compromise between (1) a short start-up delay on the one hand, and (2) caching a part of the flow, sufficient to cross an out of coverage area without service disruption, on the other hand. Depending on the network conditions (i.e. available bandwidth, terminal density, etc.), enhancement layers could be progressively added or omitted, in order to "smoothly" adapt the quality of the received flows.
Combining those different criteria, we conceived the algorithm detailed below. `BL` refers to the base layer. `EL1` represents the first enhancement layer and `EL2` the second one. `TA` means that the terminal is located in the *Transfer Area*. `HL` refers to the critical terminal cache filling-up level. The priority queues are numbered from one to six, classified with a decreasing order of priority. Let us consider a flow requested by a terminal. The three layers of that flow will have to be classified by the scheduler. Finally for each layer, the following algorithm will be iterated.

Algorithm 1 Scheduling policy in the Access Controller

```
if (Layer = BL) & (TerminalCacheLevel ≤ HL) then
    Priority Queue 1
else
    if (Layer = EL1) & (TerminalCacheLevel ≤HL) & TA then
        Priority Queue 2
    else
        if (Layer = EL2) & (TerminalCacheLevel ≤ HL) & TA then
            Priority Queue 3
        else
            if (Layer = BL) & (TerminalCacheLevel ≥ HL) & TA then
                Priority Queue 4
            else
                if TA then
                    Priority Queue 5
                else
                    Priority Queue 6
                end if
            end if
        end if
    end if
end if
```

This algorithm favors service continuity as it guarantees, with the highest priority, the delivery of the base layer to terminals whose cache is under the critical threshold. Moreover, it clearly appears that it favors flow delivery to terminals which are in transfer area. In fact, they take benefit of the high bandwidth available, which allow them to store some extra part of the flow in their cache to avoid service disruption while crossing presence areas.

4 Simulation and main results

Our solution has been evaluated in the framework of a discrete event modeling based on a Java simulator, which uses DESMO-J (Discrete event simulation framework in Java) library [11]. The simulated surface is 1 km^2. There are one network cache and six Access Points (AP). The server streaming rate is 1 Mb/s (256 kb/s for the base layer, 256 kb/s for the first enhancement layer and 512 kb/s for the second one). The simulated environment is dense urban environment like Manhattan. Each AP defines a radio cell, with a range of 50 m. A set of terminals move in a Manhattan topology, randomly crossing different radio cells and out of coverage areas, at a speed of 50 km per hour. Their cache content allows them staying out of coverage at most during 240 seconds without service disruption. The terminal may ask for any video flow in the streaming server. If this flow is already stored in the network cache, its layers are scheduled and transmitted directly from the network cache to the terminal. Else, the three layers are sent from the server, scheduled by the network cache and transmitted to the terminal, while being simultaneously stored in the network cache. The terminal can start playing the video flow since the base layer reaches the start-up threshold, which is equivalent to a playback duration of 30 seconds.
In order to evaluate our system, it seems important to point out our main goals. In fact, we need to support an important number of terminals while guaranteeing short start up delays, and service continuity. To aim at that target, we adopted a layered flow model, and we took this model into account for the design of the caching and scheduling mechanisms. The main principle was to distribute the layers depending on the terminal position and cache level. We choose to guarantee service continuity, even if we had to temporarily degrade the flow quality, by omitting some enhancement layers.
In the next sections, the performances of the designed system are evaluated, considering the start-up delay (section 4.1) and the system behavior when the terminal density increases (section 4.2).

4.1 Starting of the streaming application

In the starting of our streaming application, the network cache is empty. Let us consider a terminal, alone in the network. First, the terminal requests a video flow which is not already cached in the network cache. The flow is sent to the terminal from the streaming server, and simultaneously stored in the network cache. Here, our main goals are (1) to provide as quickly as possible the amount of data required at first to start the streaming application (*i.e.* terminal cache level = 30 s * Service Data Rate) and (2) to avoid service disruption (*i.e.* terminal cache level = 240 s * Service data rate). Therefore, we introduced a fast start-up period during which the network cache is able to retrieve the beginning of the stream two times faster than the normal service rate (Fast start-up rate = 2 Mb/s, normal service rate = 1 Mb/s). As the terminal is provided with a doubled data rate, the time necessary to start the streaming application decreases from 30 s to 15 s [1]. This fast start-up phase lasts after **240** seconds, and the flow starts to be delivered at the normal application data rate (1 Mb/s). Figure 5 illustrates the network cache content during the experiment. In this figure, the whole flow (the total of the three layers) is represented. At **240** seconds, the figure illustrates that application data rate decreases after the fast start-up phase (the graph's slope decreases). After that (at **250** seconds), the terminal decides to stop this first streaming request. At that moment, the terminal did not ended the whole flow streaming. We may notice that the flow is not totally stored in the network

cache at that moment (see figure 5). The terminal cache empties at the end of the streaming request. Nevertheless, in the network cache, the part of the flow stored is kept there. Now (at **250 seconds**), the terminal asks for the same flow, a second time. The second streaming request starts. A part of the flow is already cached in the network cache as shown in figure 5. This figure shows that, during this second streaming request, the part of the flow, remaining at the content server level, is still delivered to the network cache. At **400** seconds, the whole flow is stored in the network cache.

Figure 6 represents the evolution of the terminal cache content. In this figure, the flow layers are

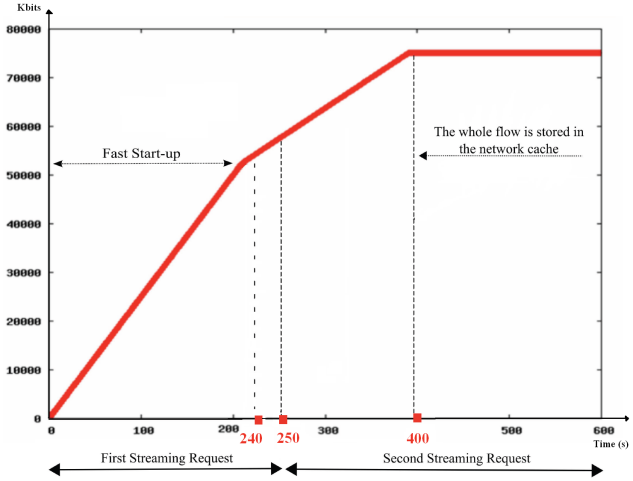

Figure 5: Network cache filling

represented separately. During the first streaming request, figure 6 shows that the terminal cache starts filling until its content reaches the playback start level (at **15** seconds). Then, the terminal starts playing the video flow at a 1 Mb/s rate, while it continues receiving the flow from the network cache at a 2 Mb/s rate. Since **240** seconds, the fast start-up period lasts and the application rate become equal to 1 Mb/s. The mobile terminal consumes its flow at the same rate it receives it, that is why we observe a plateau during this period. At **250** seconds, the terminal ends the first streaming request and empties its cache. At **250** seconds, the terminal asks for streaming the same video flow a second time. The second streaming request starts. Figure 6 shows that the base layer, and the first enhancement layer are sent at 256 kb/s, while enhancement layer 2 is transmitted at 512 kb/s. The second enhancement layer is thus received more rapidly than both the base layer and the first enhancement layer. The figure also shows that the layers are scheduled in the defined hierarchical order, which highlights the role of the scheduler.

Now, the terminal cache fills-up faster, as the flow is stored at the network cache level. As a consequence, the start-up threshold is reached more rapidly (The start-up threshold is the terminal cache level which corresponds to a playback duration of 30 seconds). Actually, the start-up threshold is reached in about 2 seconds in the second request while it is reached in about 15 seconds for the first one. This demonstrates the network cache impact during service starting.

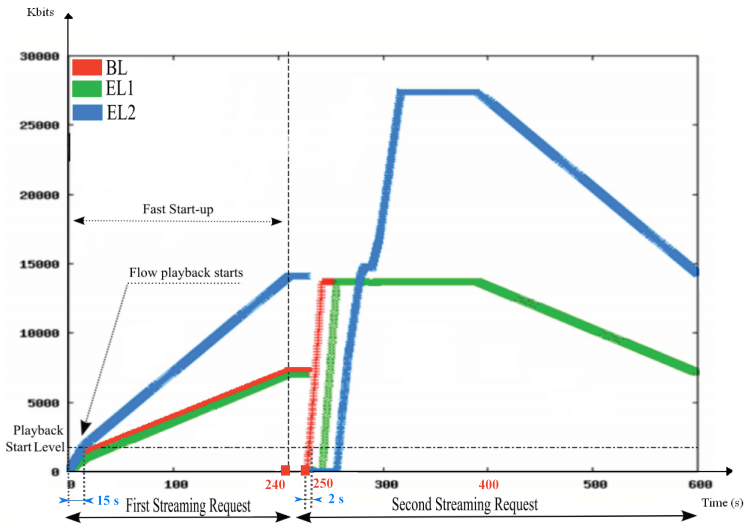

Figure 6: Terminal cache filling

We may notice that, at the beginning of the first streaming request, the network cache was empty, but when the second request starts, a part of the requested flow is already cached at the network cache. As a consequence, this part of the flow is very rapidly delivered to the terminal thanks to the high bandwidth available. Actually, as the flows are located in the network cache, they are not submitted to the bandwidth constraints at the video server level.
During the starting of a streaming request, some images might be played while only one or two layers are received. If only the base layer is received, the image is in base quality. If base layer and enhancement layer 1 are received, the quality is good. Finally, if all the layers are received, the quality is excellent. Table 1 details the percentage of images received, in each quality, by the considered terminal.

4.2 Scalability of the solution

If a terminal, alone in the network, requests a video flow already stored in the network cache, the three layers are almost immediately received in the terminal cache. As terminal density increases, reception becomes slower, because the available bandwidth is shared between users. Moreover, the flow quality may need to be degraded (by omitting enhancement layers) in order to send base layers to the whole users without service disruption. Figure 7 represents the filling-up of the cache of one terminal, for different values of terminal densities. The figure shows that the enhancement layers reception is delayed as the number of terminals increases. The idea is that the base layer of the requested flows must be sent to all terminals in order to maintain service continuity. Enhancement layers could be delayed or even omitted in order to avoid service discontinuity. Moreover, acknowl-

Table 1: Received images quality

Received images quality	
Base quality	1,6 %
Good quality	3,9 %
Excellent quality	94,4 %

edgments sent by terminals allow the network cache to avoid sending enhancement layers of images already played at the client side with base quality. Table 2 illustrates how the system progressively

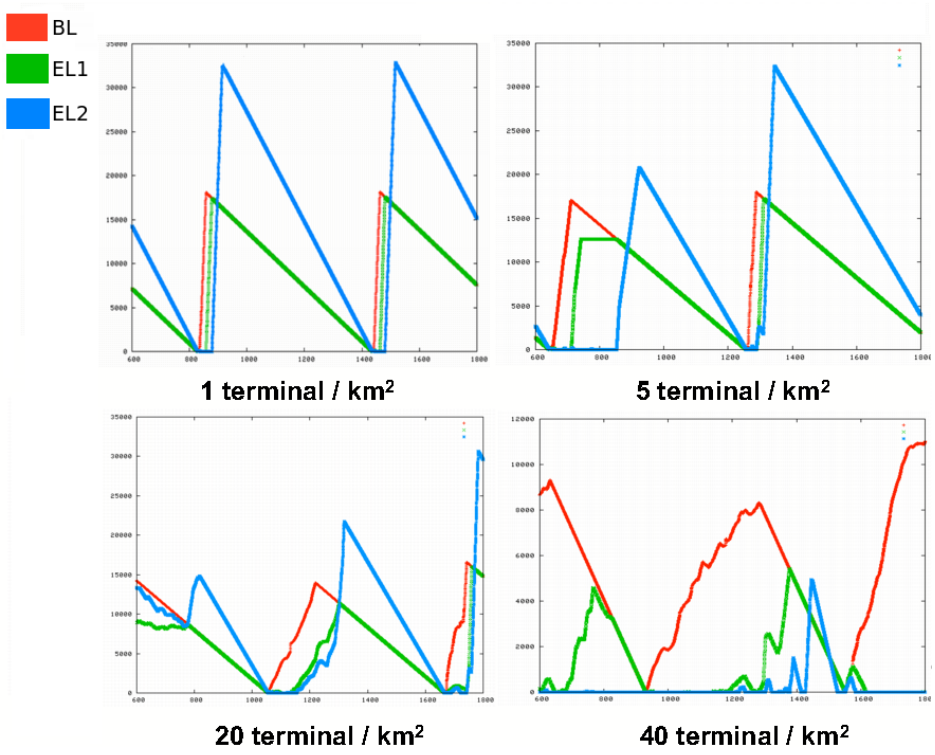

Figure 7: Terminal cache filling

degrades the received flows quality in order to maintain the streaming service continuity. Actually, the table details the evolution of the percentage of images received in base quality, in good quality and in excellent quality by one considered terminal, while terminal density increases.

Table 2: Received images quality

Received images quality						
terminals density	10	15	20	25	30	40
Base quality	4,03 %	7,15 %	2,84 %	6,99 %	19,76 %	39,86 %
Good quality	3,15 %	4,26 %	6,54 %	15,04 %	27,09 %	42,68 %
Excellent quality	92,8 %	88,57 %	90,61 %	77,95 %	53,14 %	17,45 %

5 Conclusion

We work on extending logical discontinuous coverage networks to a multitude of services. This paper addresses the streaming service. We actually tried to deliver unicast video flows to a multitude of mobile terminals, in a logical discontinuous coverage network. The main goal was to guarantee a good quality of service, especially thanks to short start-up delays and system scalability. To achieve this goal, appropriate cache and scheduling modules have been designed, taking into consideration the video flow model adopted. The network cache plays a crucial role. As it is nearer to the terminals than the origin server, start-up delays are considerably reduced. Actually, flow delivery is no more submitted to bandwith limitations at content server side. Terminal caches allow flow storing in order to avoid service disruption while being in out of coverage areas, or low throughput areas. The layered flow scheduling guarantees system scalability, thanks to the possibility of quality degradation, in order to maintain the service without any disruption. Our future work will concentrate on establishing replacement policies for the network cache, and performing new simulations while varying the scheduling algorithm.

References

[1] Antoine Luu, Marie-Line Alberi Morel, Sylvaine Kerboeuf, Ronan Ménard and Mazen Tlais and Frédéric Weis (2007), *Improving Wireless Network Capacity by Introducing Logical Discontinuous Coverage Report*, (1877), 2007.

[2] Tewari, R.,Vin, H. M., Dan, A., and Sitaram, D. (1998), *Resource-based caching for Web servers*. In Proceedings of SPIE/ACM Conf. on Multimedia Computing and Networking (MMCN 98), San Jose, CA.

[3] Sen, S., Rexford, J., and Towsley, D. (1999). *Proxy prefix caching for multimedia streams*. In Proceedings of IEEE INFOCOM 99, New York, NY.

[4] Wu, K. L., Yu, P. S., and Wolf, J. L. (2001). *Segment-based proxy caching of multimedia streams*. In Proceedings of World Wide Web Conference(WWW10), Hong Kong.

[5] Zhang, Z.-L., Wang, Y., Du, D., and Su, D. (2000). *Video staging: A proxy server- based approach to end-to-end video delivery over wide-area networks*. IEEE/ACM Transactions on Networking, 8(4): 429-442.

[6] Tang, X., Zhang, F., and Chanson, S. T. (2002). *Streaming media caching algorithms for transcoding proxies*. Proceedings of 31st International Conference on Parallel Processing (ICPP02).

[7] Kangasharju, J., Hartanto, F., Reisslein, M., and Ross, K. W. (2002). *Distributing layered encoded video through caches*. IEEE Transactions on Computers, 51(6), pp. 622-636.

[8] Y.-C. Wang, Y.-C. Tseng, and W.-T. Chen, *MR-FQ: A Fair Scheduling Algorithm for Wireless Networks with Variable Transmission Rates*, Simulation: Transactions of The Society for Modeling and Simulation International, accepted, 2005. (SCI)

[9] REJAIE, R., AND REIBMAN, A. *Design Issues for Layered Quality Adaptive Internet Video Playback*. In Proc. of the Workshop on Digital Communications (Taormina, Italy, September 2001).

[10] Joe Yuen, Kam-Yiu Lam, and Edward Chan, *A Fair and Adaptive Scheduling Protocol for Video Stream Transmission in Mobile Environment*, in Proceedings of IEEE International Conference on Multimedia and Expo, Lausanne, Switzerland, August 2002.

[11] URL : http://asi-www.informatik.uni-hamburg.de/desmoj/